Coupling Two Constraint-Based Systems Into an On-line Façade-layout Configurator

Andrés F. Barco[1] and **Élise Vareilles**[1] and **Paul Gaborit**[1] and
Jean-Guillaume Fages[2] and **Michel Aldanondo**[1]

Abstract.
We present the coupling of two constraint-based environments into an on-line support system for façade-layout configuration in the context of building renovation. The configuration consist on the definition and allocation of a set of rectangular parameterizable panels over a façade surface. The coupling allows to solve two configuration tasks while gaining efficiency and modularity. First, it allows to configure a set of questions relating the renovation model needed to determine limits for panels' size and panels' weight. Second, it allows to configure a constraint satisfaction model for each of the façades to renovate. Two constraint-based systems handle the filtering of incompatible values and the generation of layout plans in a web-service setup. The first service performs initial filtering to set panels' limits, based on the questionnaire, using a constraint filtering engine called `CoFiADe`. The second service uses several façade-layout configuration algorithms, using as underlying engine the constraint solver `Choco`, to generate compliant layout-plan solutions. We show that by dividing filtering and search, and by coupling the two constraint-based systems, we gain modularity and efficiently as each service focuses on their own strengths. Services executing tasks may be hosted in different network-nodes and thus may be seen as independent communicating agents.

1 Preliminaries

Constraint-base façade-layout configuration. Product configuration refers to the task of building a target product using predefined components, respecting requirements from customers and following some rules that shape a correct configuration [26, 29]. This task have been increasingly supported by intelligent systems given the complexity and size of relations within a single product. For instance, configuring a computer from memories, buses, cards and so on, involves a large number of possibilities and solutions for the user. The possible numbers of outputs for a configuration is in relation to the number of components and relations within the product, and is inversely proportional to the number of rules that restrict combinations. We call these kind of problems *combinatorial problems*.

A particular scenario on product configuration arises from the context of building thermal renovation as an effort to reduce current energetic consumption levels [5, 6, 19]. Here, the problem lies on the configuration of rectangular parameterizable panels, and their attaching devices called fasteners, that must be allocated over the façade surface in order to provide an insulation envelope [8, 14, 28]. The problem is also known as layout synthesis or space planning. A configuration solution is a plan which satisfies optimization criteria and a set of constraints (such as geometrical, weight or resources constraints) provided by users and the façade itself. As part of the product configuration family problems, an instance of façade-layout configuration problem has a huge search space that depends on the size of the panels and the elements on the façade, such as windows, doors and supporting areas (see Figure 1). In consequence, to solve this configuration problem, we choose to rely on a technique from artificial intelligence and operation research called constraint satisfaction problems [15, 18].

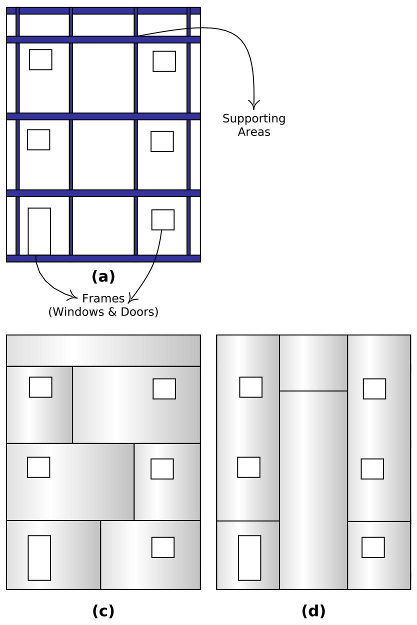

Figure 1: A façade is a rectangular surface with supporting areas, windows and doors. Layout-plans solutions made out configured rectangular panels.

Constraint satisfaction problems (CSPs) are conceived to allow the end-user to state the logic of the computation rather than its flow. For example, in the context of scheduling, instead stating a set of steps to avoid tasks overlapping, the user declares *"for any pair of tasks they must not overlap"*. The user may do so by stating a) variables representing elements of the problem, b) a set of potential values associated to each variable and c) relations over the stated variables also known as constraints [12, 18]. Variables may have different do-

[1] Université de Toulouse, Mines d'Albi, Route de Teillet Campus Jarlard, 81013 Albi Cedex 09, France, email: abarcosa@mines-albi.fr
[2] COSLING S.A.S., 2 Rue Alfred Kastler, 44307 Nantes Cedex 03, France

main representation such as integer or boolean, and relation among variables may be expressed in different ways such as compatibility tables and first order formulas. Solving a CSP means finding an assignment of values for each variable in such a way that all constraints are satisfied.

It turns out that constraint technology fits neatly in the constrained nature of product configuration and layout synthesis. On one hand, the knowledge (constraints) that restrict possible configuration of elements (variables) is easily modeled under the declarative framework of constraint satisfaction problems. On the other hand, constraint-based configurators are able to presents different solutions to users, often optimal, even when they do not provide all configuration parameters leaving their preferences unknown.

Now, for constraint-based implementations we differentiate between two related concepts: Solving a problem and filtering (narrow) possibilities. Whereas solving a problem involves a robust inference engine and search strategies, filtering algorithms work, in essence, on how to efficiently remove variable values that are restricted by the established relations, i.e., the constraints [2, 4]. In the problem at hand both the filtering and the search take part; filtering to set panel's size and weight limits and, search in order to generate compliant layout plans.

Related work. Layout configuration techniques have been used within different contexts and scenarios. For instance, finding solutions for room configurations [30], apartment layouts [16] and activities within a business office [13]. Also, some tools have been implemented using different approaches, here we name a few of them. For example, in [25] Shikder et al. present a prototype for the interactive layout configuration of apartment buildings including design information and an iterative design process. In [3] is introduced *Wright*, a constraint-based layout generation system that exploits disjunctions of constraints to manage the possibilities on positioning two-dimensional objects in a two-dimensional space. Another system, *Loos* [9], is able to configure spaces using rectangles that can not be overlapped but that may have holes. It uses test rules applied by steps to the rectangles in order to reach a good configuration based on its orientation and relation with other rectangles. The same authors have developed *Seed* [10, 11]: A system based on *Loos* used for early stages on architectural design. The system *Hegel* [1] (for Heuristic Generation of Layouts) is yet another space planning tool that simulates human design based on experimental cases. Finally, Medjdoub et al. presents in [17] the system *Archiplan* which integrates geometrical and topological constraints to apartment layout planning.

Regardless the considerable number of applications on layout configuration, our problem include three characteristics never considered simultaneously: Its deals with the allocation of an *unfixed number of rectangular* panels that must not overlap, frames (existing windows and doors) must be *overlapped* by one and only one panel, and façades have specific areas providing certain *load-bearing capabilities* that allow to attach panels. Thus, as far as we know, no support system nor design system is well-suited for addressing such particularities. Also, most systems are desktop-oriented and not web-oriented, making difficult to adapt new requirements and functionalities as they need new versions to be released.

Contribution and structure. Traditional general purpose constraint solvers, such as `Gecode` [24], `Choco` [20] and the finite domain module of `Oz` [23], make clear distinction between filtering and search [21]. These environments provide methods for invoking constraint propagation, i.e., executing the filtering algorithm of constraints, and methods for invoking search for one solution, several solutions or optimal ones [22]. Nonetheless, to the best of our knowledge, studies focusing on constraint-based configuration involving filtering and search does not make clear distinction between these concepts and their implementation focuses on the search. This means that solutions exploits the capabilities of constraint solvers as black-box environments, typically creating a dedicated search heuristic for the problem.

Our goal is two-fold. First, we propose an architecture that divides initial filtering and consequent search for constraint-based product configuration. The architecture allow us to solve two configuration tasks; configure a set of questions relating the renovation model needed in the renovation process and needed to determine limits for panels' size and panels' weight and; configure a constraint satisfaction problem for each of the façades to renovate. In a second time, we present an on-line support system, and formalize its behavior, for the problem of façade-layout configuration. The architecture, implemented in the on-line system, couples two different constraint-based technological tools to gain efficiency and modularity. We use façade-layout configuration to illustrate our method as it is the goal of the project we are into, but our results can be adapted to deal with different kind of products.

The paper is divided as follows. A brief description of the industrial scenario and elements is presented in Section 2. In section 3, we introduce details of the two configuration tasks performed by the support system. The service oriented architecture, along with details of the constraint-services' behavior, is presented in Section 4. In Section 5, we discuss the benefits of the tasks division and coupling of the constraint systems. Some conclusions are drawn in Section 6. A bibliography is provided at the end of the document.

2 Renovation Modus Operandi

The problem at hand deals with the configuration of rectangular panels and their arrangement over a façade surface. In order to start the configuration process, specific information have to be extracted from a set of spatial entities. The renovation is carried out on *façades* that are part of a given building, *buildings* that are part of a given block, and *blocks* that are part of a given *working site*. Each of these spatial entities have geometrical and structural properties and may have different environmental conditions that must be taken into account for the layout-plan definition.

Information about spatial entities is acquired by the support system by means of an (JSON) input file describing all geometrical and structural properties, and by means of a web-based questionnaire for each spatial entity in the input file. Thus, if the input file contains information for one working site with two blocks, each block with one building and three façades in each building, the user answers to eleven questionnaires (one for the working site, two for the blocks, two for the buildings and six for the façades). After questionnaire completion, the lower bound and upper bound for panels' size and panel's weight have been deduced. Also, given that several instances for façades are need to be solve, at the end of the questioning stage the systems creates a constraint satisfaction model for each façade using the inputed information and the deduced limits for panels' size and weight. Here, each constraint satisfaction model instance is parameterized according with the façade information (e.g. environmental conditions) and the particular deduced panels' limits.

Lets consider the information flow from the user perspective. Figure 2 presents the representative actions made by the user and the responses by the on-line support system. It also illustrates the fact that to the end-user should be transparent all the configuration process. The complete sequence of the configuration goes as follows.

Step 1:- The user uploads a file containing the geometry and structural specification of spatial entities. The support system stores information in a data base.

Step 2:- The filtering service presents a questionnaire for each of the spatial entities in the input file.

Step 3:- The user answers the questions (leaving in blank the questions he does not know the answer).

Step 4:- Using the information about spatial entities (database) and their environmental/user conditions (user answers) the system deduce lower and upper bounds for panels' size and panels' weight by using the filtering service.

Step 5:- If a manual configuration is desired, the user draws each panel on the clients GUI. Each panel is assured to be consistent with the problem requirements by sending its information to validate into the solving service (the validator module).

Step 6:- If a semi-automatic configuration is desired, the user draws some panels and then asks the solving service to finish the configuration.

Step 7:- If an automatic configuration is desired, the user asks the solving service to provide compliant layout solutions.

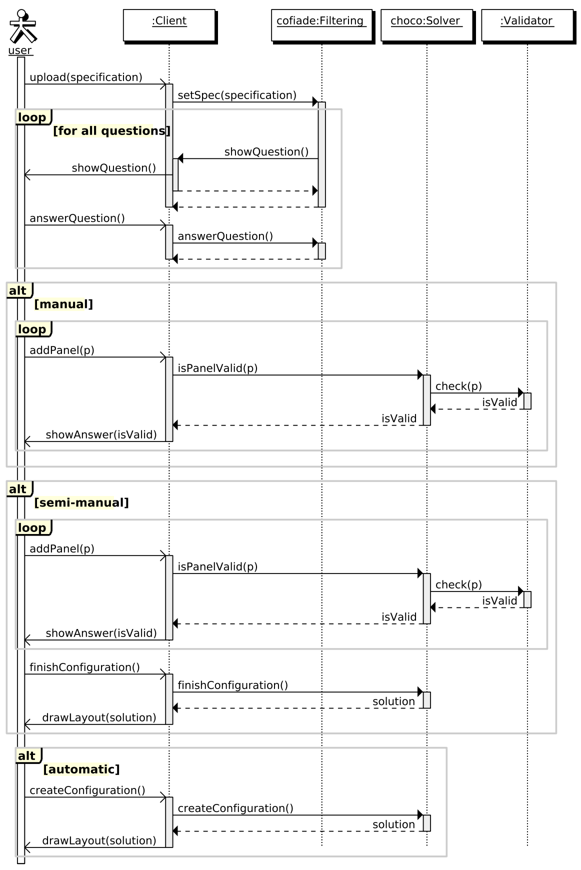

Figure 2: Sequence diagram for on-line support system.

It is worth mentioning the consistency among the different steps: Information at each level is propagated downwards and is never propagated upwards (we will further study this along the document). Also, of major importance is the fact that each façade may have its own panels' lower and upper bounds and thus solving a given façade is done with a particular set of arguments.

3 Support System Configuration Tasks

In this section we present the two configuration tasks within the support system: The configuration of a questionnaire to be filled by the end-user and, the configuration of a constraint satisfaction model for each façade to renovate used as input for layout-plans generation.

3.1 The Questionnaire

The renovation includes four spatial entities, namely, working site, block, building and façade, and some configurable components, namely, panels and fasteners (fasteners are devices to attach panels onto the façades). A hierarchical view is presented in Figure 3. Once the input file has been read by the support system, it can proceed by configuring a set of questions for each spatial entity in the file. Then, after the user answer the questionnaires, the system configures, i.e., deduces, the limits for panels' size and panels' weight for *each façade*. The questionnaires ask the following information.

Working site. This is the bigger spatial division in the renovation. It is commonly referred by a name and is well-know by the community. Values provided by the user are:

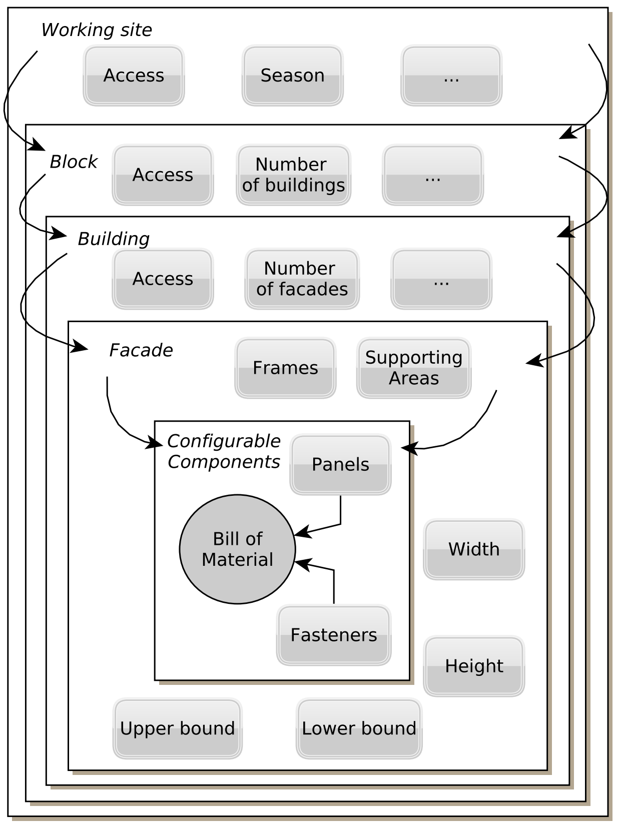

Figure 3: Spatial entities and arguments for renovation.

- Number of blocks in the working site? Number.
- Working site is in a windy region? $\{$yes, no$\}$
- Season in which the on-site work will take place? $\{$summer, fall, winter, spring$\}$
- Target for cost? Euros.
- Target for performance? $W.m^{-2}.K^{-1}$
- Obstacles presence? $\{$yes, no$\}$
- Accessibility to the working site? $\{$easy, medium, hard$\}$
- Panel's width ($\underline{w_{ws}}$) and height ($\underline{h_{ws}}$) lower bound? $[0, \infty]$
- Panel's width ($\overline{w_{ws}}$) and height ($\overline{h_{ws}}$) upper bound? $[0, \infty]$
- Panel's maximum weight ($\overline{we_{ws}}$)? $[0, \infty]$

Block. Is a set of buildings which are usually attached by a common wall. Values provided by the user are:

- Number of buildings in the block? Number.
- Obstacles presence? $\{$yes, no$\}$
- Accessibility to the block? $\{$easy, medium, hard$\}$
- Panel's width ($\underline{w_{bl}}$) and height ($\underline{h_{bl}}$) lower bound? $\underline{w_{bl}} \in [\underline{w_{ws}}, \overline{w_{ws}}]$ and $\underline{h_{bl}} \in [\underline{h_{ws}}, \overline{h_{ws}}]$
- Panel's width ($\overline{w_{bl}}$) and height ($\overline{h_{bl}}$) upper bound? $\underline{w_{bl}} \in [\underline{w_{ws}}, \overline{w_{ws}}]$ and $\underline{h_{bl}} \in [\underline{h_{ws}}, \overline{h_{ws}}]$
- Panel's maximum weight ($\overline{we_{bl}}$)? $[0, \overline{we_{ws}}]$

Building. Is the actual place where apartment are arranged and is the host of several façades. Values provided by the user are:

- Number of façades in the building? Number.
- Obstacles presence? $\{$yes, no$\}$
- Accessibility to the block? $\{$easy, medium, hard$\}$
- Panel's width ($\underline{w_{bg}}$) and height ($\underline{h_{bg}}$) lower bound? $\underline{w_{bg}} \in [\underline{w_{bl}}, \overline{w_{bl}}]$ and $\underline{h_{bg}} \in [\underline{h_{bl}}, \overline{h_{bl}}]$
- Panel's width ($\overline{w_{bl}}$) and height ($\overline{h_{bl}}$) upper bound? $\underline{w_{bg}} \in [\underline{w_{bl}}, \overline{w_{bl}}]$ and $\underline{h_{bg}} \in [\underline{h_{bl}}, \overline{h_{bl}}]$
- Panel's maximum weight ($\overline{we_{bg}}$)? $[0, \overline{we_{bl}}]$

Façade. Maybe seen as a big wall, but is in fact a composition of apartment along with its doors, windows and so on. Values provided by the user are:

- Obstacles presence? $\{$yes, no$\}$
- Accessibility to the block? $\{$easy, medium, hard$\}$
- Type of attaching device: $\{$bottom, top, lateral$\}$
- Panel's width ($\underline{w_{fc}}$) and height ($\underline{h_{fc}}$) lower bound? $\underline{w_{fc}} \in [\underline{w_{bg}}, \overline{w_{bg}}]$ and $\underline{h_{fc}} \in [\underline{h_{bg}}, \overline{h_{bg}}]$
- Panel's width ($\overline{w_{bl}}$) and height ($\overline{h_{bl}}$) upper bound? $\underline{w_{fc}} \in [\underline{w_{bg}}, \overline{w_{bg}}]$ and $\underline{h_{fc}} \in [\underline{h_{bg}}, \overline{h_{bg}}]$
- Panel's maximum weight ($\overline{we_{fc}}$)? $[0, \overline{we_{bg}}]$

This information collection has two specific goals. On the one hand, it will provide details about renovation aspects, such as the targeted performance, that are needed in the configuration process. On the second hand, it provides upper bound for panel's size and panel's weight. Indeed, given the manufacturing, environmental, accessibility and even weather conditions, the size of panels composing the layout plan are limited as well as their weight. Thus, the aforementioned inputs have direct impact over configurable components and are defined by their compatibility relations.

3.2 One FaçAde One CSP

One critical aspect of the support system is the ability to configure a constraint satisfaction model for each façade to renovate. This is important because, first, each façade has (potentially) different size, number of windows, supporting areas etc. Possible positions for panels, for instance, must lie between zero and the façade width and height. Simply put, each façade has its own configuration parameters used in the constrain satisfaction model and in the layout generation process. And second, each façade may have different accessibility conditions, obstacles or even user preferences. Thus, panels' size limits, as well as their weight, are constrained by the specific conditions of the façade and not only by the conditions of the working site, block or building.

When configuring these CSP instances it is important to conserve downwards consistency. Downwards consistency refers to the fact that information on higher level of the renovation are is propagated to the inferior levels, i.e., working site $\rightarrow$ blocks $\rightarrow$ buildings $\rightarrow$ façades, but it can not propagate upwards. As an example consider only accessibility conditions, obstacles presence and panels' size limits, for the specification in Figure 4.

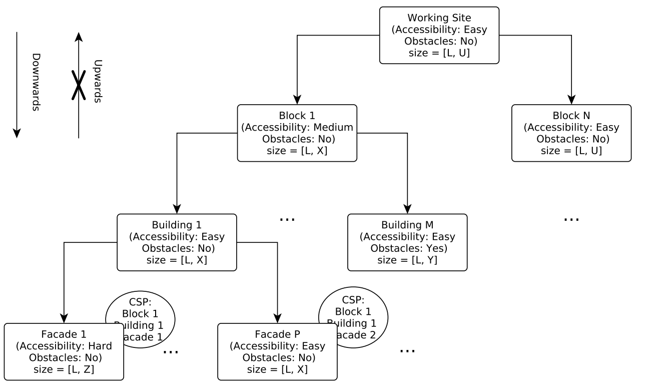
Downwards
Upwards
Working Site (Accessibility: Easy Obstacles: No) size = [L, U]
Block 1 (Accessibility: Medium Obstacles: No) size = [L, X]
Block N (Accessibility: Easy Obstacles: No) size = [L, U]
Building 1 (Accessibility: Easy Obstacles: No) size = [L, X]
Building M (Accessibility: Easy Obstacles: Yes) size = [L, Y]
CSP: Block 1 Building 1 Facade 1
CSP: Block 1 Building 1 Facade 2
Facade 1 (Accessibility: Hard Obstacles: No) size = [L, Z]
Facade P (Accessibility: Easy Obstacles: No) size = [L, X]
...

Figure 4: Downwards consistency among entities.

Note that inferior entities on the hierarchy inherit values from superior levels. But, it is not the case that information on superior levels should be consistent with information on inferior levels. In Figure 4, for instance, façade 1 has a hard accessibility condition and thus the upper bound for panels' size is reduced to a given Z. This upper bound is not propagated upwards to the building 1; it conserves its inherited value X. Consequently, façade 2 will inherit the value of X as no further reduction is needed for their panels configuration. Naturally, it is the case that $Z \subset X \subset U$. Using this information a CSP is configured for each façade to renovate.

4 Support System Constraint Services

In order to divide configuration tasks we divide the support system in two services that may be implemented in different servers. In essence, information about the renovation, entered by means of the input file and the questionnaire, is filtered by means of the first service called *Filtering Service*. Then, the second service called *Solving Service*, upon user request, uses its configuration algorithms to provide complaint layout-plans solutions. Figure 5 presents the architecture of the on-line support system.

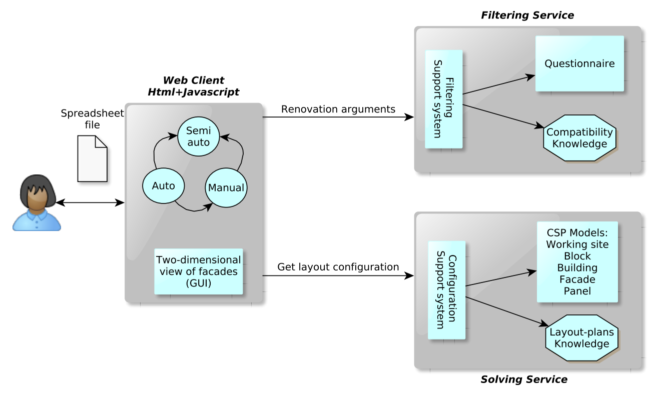

Figure 5: Service-based architecture for on-line configurator.

In order to give a clear understanding on how the services works, let us describe the input and output in a formal way. For each of the services the input is a tuple of the form $\langle \mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}(V)\rangle$ with $|\mathcal{V}|= |\mathcal{D}(\mathcal{V})|$ and

- $\mathcal{SPEC} = \langle \mathcal{WS}, \mathcal{BK}, \mathcal{BG}, \mathcal{FAC}\rangle$; $\mathcal{WS}$ variables describing the working site, $\mathcal{BK}$ variables describing blocks, $\mathcal{BG}$ variables describing buildings and $\mathcal{FAC}$ variables describing façades.
- $\mathcal{V} = \langle \mathcal{P}, \mathcal{FA}\rangle$; $\mathcal{P}$ variables describing a single panel and $\mathcal{FA}$ variables describing a single fastener.
- $\mathcal{D}(\mathcal{V}) = \langle \mathcal{D}(\mathcal{P}), \mathcal{D}(\mathcal{FA}), \rangle$; domain for each one of the variables in $\mathcal{V}$.
- $\mathcal{C}(\mathcal{V})$ a set of constraints over variables in $\mathcal{V}$.

Information in $\mathcal{SPEC}$ describes only properties of the spatial entities such as the number, sizes, positions, etc. Variables in $\mathcal{V}$ and $\mathcal{D}(\mathcal{V})$, on the other hand, are manufacturer depended and includes size and position of panels and fasteners, and the initial domains which depends on the manufacturing process. Constraint in $\mathcal{C}(\mathcal{V})$ are extracted from the problem domain by an expert and are different in each service.

4.1 Filtering Service

4.1.1 Mapping

The filtering service is in charge of removing domain values from elements in $\mathcal{D}(\mathcal{V})$ that are not allowed by the established constraints. Here, constraints $\mathcal{C}(\mathcal{V})$ describe valid combination among different parameters in $\mathcal{SPEC}$ and variables in $\mathcal{D}(\mathcal{V})$. We denote this set of constraints $\mathcal{C}_f(\mathcal{V})$ to distinguish them from the ones used on the solving service. These constraints are formalized as compatibility tables (presented in next section). Formally, the filtering is a mapping $\mathcal{M}$ from variables and domains to domains

$$\mathcal{M}(\mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}_f(\mathcal{V})) \rightarrow \mathcal{D}'(\mathcal{V}) \qquad (1)$$

The result $\mathcal{D}'(\mathcal{V})$ contains the new domain for panels and fasteners, where $\mathcal{D}'(\mathcal{V}) \subseteq \mathcal{D}(\mathcal{V})$.

As stated previously, the initial filtering has as goal setting domains for configurable components and takes spatial entities information and constraints to do so. In our on-line support system we use the `CoFiADe` [27] system to perform this filtering. Several reasons support our choice. First, the system is already on-line, making it usable in no time. Second, it is well conceived for supporting decision-making processes. And third, it uses efficient compatibility tables for domain pruning; applying a given compatibility table is made in constant time $\mathcal{O}(1)$.

4.1.2 Compatibility Knowledge

Configurable components of the renovation are panels and fasteners to attach panels.

Panels. Configurable by fixing their width, height, weight and position over the façade.
Fasteners. Configurable by fixing its length and setting its type {bottom, top, lateral}.

The following compatibility tables, presented from Table 1 to Table 8, show the allowed combination between user's input values and configurable components values.

4.2 Solving Services

4.2.1 Search

The second service in the support system is in charge of layout-plans generation. The system uses several algorithms to generate layout plans but, although their behavior are quite different, their semantic remains the same.

Now, while information of $\mathcal{SPEC}$ and $\mathcal{V}$ are the same as the filtering services, it is not the case for domains and constraints. To differentiate them lest call the input domains $\mathcal{D}_s(\mathcal{V})$ and the constraints $\mathcal{C}_s(\mathcal{V})$. Intuitively, variable domains $\mathcal{D}_s(\mathcal{V})$ are provided by the mapping of the filtering service, i.e.,

$$\mathcal{M}(\mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}_f(\mathcal{V})) = \mathcal{D}'(\mathcal{V}) = \mathcal{D}_s(\mathcal{V}) \qquad (2)$$

where $\mathcal{D}(\mathcal{V})$ is the initial variable domain of the problem. Constraints in $\mathcal{C}_s(\mathcal{V})$ are expressed as first order formulas and express, not compatibility among elements but, requirements for valid layout plans (see next section for a description of these constraints). Both resolution approaches implemented at the solving service respect all constraints; the first approach resolving conflicts at positioning each panel (greedy fashion) whereas the second approach uses the open constraint programming environment `Choco` [20] to explore the solution space.

The output of the server's process is a set of layout-plan solutions. Formally, the server's process is a function of the form

$$\mathcal{F}(\mathcal{SPEC}, \mathcal{V}, \mathcal{D}_s(\mathcal{V}), \mathcal{C}_s(\mathcal{V}), \mathcal{H}) = \langle \mathcal{X}, \mathcal{Y}, \mathcal{DX}, \mathcal{DY}\rangle \qquad (3)$$

where $\mathcal{X}$ and $\mathcal{Y}$ represent the origin of coordinates for each panel in the solution, and $\mathcal{DX}$ and $\mathcal{DY}$ the width and height, respectively, for each panel in the solution. Additionally, the function is parameterized by an heuristic $\mathcal{H}$ stating the way the solution space is explored. Available strategies are greedy and depth-first search.

Table 1: Relation C_1 between environmental conditions of spatial entities and panel's size, where α and β are upper-bounds for panel's width and height, respectively, when constrained by environmental conditions.

C_1

Wind	Panel's size
yes	$(w_p \leq \alpha) \wedge (h_p \leq \beta)$
no	$\varnothing$

Table 2: Relation C_2 season that on-site work will take place and and panel's size, where θ and τ are upper-bounds for panel's width and height, respectively, when constrained by the season.

C_2

Season	Panel's size
Summer $\vee$ Spring	$\varnothing$
Fall $\vee$ Winter	$(w_p \leq \theta) \wedge (h_p \leq \tau)$

Table 3: Relation C_3 between obstacles in spatial entities and panel's size, where ϕ and σ are upper-bounds for panel's width and height, respectively, when constrained by the presence of obstacles.

C_3

Obstacles	Panel's size
yes	$(w_p \leq \phi) \wedge (h_p \leq \sigma)$
no	$\varnothing$

Table 4: Relation C_4 between accessibility of spatial entities and panel's size, where λ and π are upper-bounds for panel's width and height, respectively, when constrained by medium level accessibility conditions, and Λ and Π are upper-bounds for panel's width and height, respectively, when constrained by hard level accessibility conditions.

C_4

Accessibility	Panel's dimensions
easy	$\varnothing$
medium	$(w_p \leq \lambda) \wedge (h_p \leq \pi)$
hard	$(w_p \leq \Lambda) \wedge (h_p \leq \Pi)$

Table 5: Relation C_5 between renovation cost and panel's insulation: It illustrates the fact that the quality of the insulation depends on the user budget.

C_5

Cost	Panel's insulation
< 50000	low
[50000,100000]	medium
≥ 100000	high

Table 6: Relation C_6 between desired performance and panel's insulation: It illustrates the fact that the quality of the insulation depends on the desired final energetic performance.

C_6

Performance	Panel's insulation
< 25	high
[25,50]	medium
≥ 50	low

Table 7: Relation C_7 between panel's weight and fasteners' positions.

C_7

Weight	Position Fasteners
< 500	$\varnothing$
[500,1000]	{top,bottom}
>1000	bottom

Table 8: Relation C_8 between fasteners' position and number of fasteners.

C_8

Position fasteners	# fasteners
{top,bottom}	[2,4,6]
laterals	[4,6]

4.2.2 *Layout knowledge*

Let F denote the set of frames and S the set of supporting areas. Let $o_{e.d}$ and $l_{e.d}$ denote the origin and length, respectively, of a given entity e in the dimension d, with $d \in [1, 2]$. For instance, $o_{fr.1}$ denotes the origin in the horizontal axis and $l_{fr.1}$ denotes the width of frame fr. Additionally, lb_d and ub_d denote the length lower bound and length upper bound, respectively, in dimension d for all panels.

Each panel is described by its origin point w.r.t. the façade origin and its size. For convenience, lets assume that $\mathcal{P}$ is the set of panels composing the layout-plan solution. Then, each $p \in \mathcal{P}$ is defined by $\langle o, l \rangle$ where

- $o_{p.d} \in [0, o_{fac.d}]$ is the origin of panel p in dimension d.
- $l_{p.d} \in [lb_{p.d}, ub_{p.d}]$ is the length of panel p in dimension d.

The following six constraints express the relations among panels, and panels and façade that must respect a layout solution.

(a) *Manufacturing and transportation limitations constrain panel's size with a give upper bound* ub *in one or both dimensions.*

$$\forall_p \in P \; l_{p.d} \leq ub_d$$

(b) *(diffN) For two given panels p and q there is at least one dimension where their projections do not overlap.*

$$\forall_p \in P, \forall_q \in P, p \neq q, \exists d \in [1, 2] \mid$$
$$o_{p.d} \geq o_{q.d} + l_{q.d} \vee o_{q.d} \geq o_{p.d} + l_{p.d}$$

(c) *A given panel p must either be at the façade edge or ensure that enough space is left to fix another panel.*

$$\forall_p \in P \; o_{p.d} + l_{p.d} \leq l_{fac.d} - lb_k \vee o_{p.d} + l_{p.d} = l_{fac.d}$$

(d) *Each frame over the façade must be completely overlapped by one and only one panel. Additionally, frames' borders and panels' borders must be separated by a minimum distance denoted by* Δ.

$$\forall_f \in F, \exists p \in P \mid$$
$$o_{p.d} + \Delta \leq o_{f.d} \wedge o_{f.d} + l_{f.d} \leq o_{p.d} + l_{p.d} + \Delta$$

(e) *The entire façade surface must be covered with panels.*

$$\sum_{i \in P} \prod_{d \in [1,2]} (o_{i.2} + l_{i.2}) = \prod_{d \in [1,2]} l_{fac.d}$$

(f) *Panels' corners must be matched with supporting areas in order to be properly attached onto the façade.*

$$\forall_p \in P, \exists s \in S \mid o_{s.d} \leq o_{p.d} \vee o_{p.d} + l_{p.d} \leq o_{s.d} + l_{s.d}$$

5 (De)Coupling Benefits

In order to efficiently couple two constraint-based systems it is necessary to assign disjoint tasks for them. In our approach, we divide initial filtering and solving in two different services. Benefits for this tasks division are rather simple.

On the one hand we apply the well-known principle *Divide and conquer*. In our on-line system this principle allow us to add or remove variables, domains and questions in the filtering service, i.e., by means of adding or removing compatibility tables. In addition, as we use CoFiADe, we may mix different variable representation as integer domains, continuous domains and symbolic domains whereas in most constraint systems mixing variable domains is not allowed or is not efficient enough. For instance, given the reduced number of constraints for continuous domains in Choco, the representation have to be changed to integer domains which in consequence involve additional and time consuming efforts.

On the other hand, as a benefit of tasks division, we improve performance by avoiding the use of binary equalities and binary inequalities constraints whose computational time is $\mathcal{O}(n * m)$, where n and m are the number of values in the domain of the two variables involved in the constraint. Thus, at the moment of finding solutions, the underlying constraint solver, in our case Choco, propagates and applies search using only those constraints defining a layout plan.

Regarding the performance the two configuration tasks must be studied separately. As commented before, applying a given compatibility table in the filtering service is made in constant time. Thus, the time involved in the filtering service depends on the questionnaire that depends on the number of buildings, facades and so on. On the solving service, by contrast, the performance depends on the underlying filtering and search provided by Choco. Execution over façades with size 40×10 meters, 50×12 meters and 60×15 meters takes between one and two seconds. The use of a dedicated heuristic that exploits the problem structure allows to reach such good performance.

The decoupling of tasks, and coupling of two constraint-based systems, is supported by the underlying declarative model. Indeed, the monotonic properties of constraint-based systems make it possible to add information (constraints) on one system without loosing any solution on the other system. Thus, the declarative view of constraint satisfaction make it possible to handle services as independent communication agents.

6 Conclusions

The aim of this paper has been to introduce an architecture of constraint-based product configuration that couples two constraint-based systems. We have presented an architecture that divided initial variable domain filtering and search space exploration. The method divide and conquer allow us to make straightforward adaptations to each service separately. Our approach have been applied to façade-layout configuration and implemented in an on-line support system. For this particular scenario we have

1. Formalize each service behavior and the relation among them.
2. Presented the constraints, expressed in compatibility tables and carried out by the CoFiADe system, for initial filtering.
3. Presented the constraints, expressed as first order formulae and carried out by Choco constraint programming environment, that are used to generate compliant layout solutions.
4. Show how to solve the configuration tasks by coupling the two constraint-based systems.
5. Show that consistency and integrity of solutions are straightforward modeled and implemented thanks to the monotonic properties of constraint satisfaction problems.
6. Show that the underlying coupling and communication methods are transparent for the user (it only interacts with a friendly web-based interface).

This work is part of a project on buildings thermal renovation assisted by intelligent systems. A configuration problem arise in this context: Configuring a specific set of rectangular panels with respect to create a building envelope. As the industrial scenario evolves the support system must be able to adapt to new requirements. Thus, strategic directions for our work are three-fold. On the one hand, improve each service by adding new constraints. On the second hand, definition of an API that allow us to replace the underlying processes in each service without loosing solutions. For instance, the solving service may be replaced by the same implementation of the model but using a different constraint solver (e.g., Gecode [24], ILOG CPLEX CP [7]). We consider that a good support system must be robust enough to allow such adaptations. Finally, a consistent benchmark must be carried on in order to compare the performance when dividing configuration tasks and when they are executed by the same service/constraint system.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the TBC Générateur d'Innovation company, the Millet and SyBois companies and all partners in the CRIBA project, for their contributions on recollecting buildings renovation information. Special thanks to the referees for their comments and to Philippe Chantry from École des Mines d'Albi for his contribution to the on-line system graphical interface and additional abstractions.

REFERENCES

[1] Ö. Akin, B. Dave, and S. Pithavadian, 'Heuristic generation of layouts (hegel): based on a paradigm for problem structuring', *Environment and Planning B: Planning and Design*, **19**(1), pp. 33 – 59, (1992).

[2] R. Barták, 'Constraint Programming: In Pursuit of the Holy Grail', in *Proceedings of the Week of Doctoral Students (WDS)*, (June 1999).

[3] Can A. Baykan and Mark S. Fox, 'Artificial intelligence in engineering design (volume i)', chapter WRIGHT: A Constraint Based Spatial Layout System, 395–432, Academic Press Professional, Inc., San Diego, CA, USA, (1992).

[4] S. C. Brailsford, C. N. Potts, and B. M. Smith, 'Constraint satisfaction problems: Algorithms and applications', *European Journal of Operational Research*, **119**(3), pp. 557 – 581, (1999).

[5] The Energy Conservation Center, *Energy Conservation Handbook*, The Energy Conservation Center, Japan, 2011.

[6] U.S. Green Building Council, *New Construction Reference Guide*, 2013.

[7] IBM IBM ILOG CPLEX. Ibm ilog cplex optimization studio cp optimizer users manual, 2014.

[8] M. Falcon and F. Fontanili, 'Process modelling of industrialized thermal renovation of apartment buildings', *eWork and eBusiness in Architecture, Engineering and Construction*, 363–368, (2010).

[9] U. Flemming, 'Knowledge representation and acquisition in the LOOS system', *Building and Environment*, **25**(3), 209 – 219, (1990).

[10] U. Flemming, C.A. Baykan, R.F. Coyne, and M.S. Fox, 'Hierarchical generate-and-test vs constraint-directed search', in *Artificial Intelligence in Design '92*, eds., J.S. Gero and Fay Sudweeks, 817–838, Springer Netherlands, (1992).

[11] U. Flemming and R. Woodbury, 'Software environment to support early phases in building design (seed): Overview', *Journal of Architectural Engineering*, **1**(4), 147–152, (1995).

[12] Esther Gelle and Rainer Weigel, 'Interactive configuration based on incremental constraint satisfaction', in *IFIP TC5/WG 5.2 Workshop Series Knowledge-Intensive CAD*, pp. 117–126, Helsinki, Finland, (1995).

[13] M. M. D. Hassan, G. L. Hogg, and D. R. Smith, 'Shape: A construction algorithm for area placement evaluation', *International Journal of Production Research*, **24**(5), pp. 1283–1295, (1986).

[14] Bjrn Petter Jelle, 'Traditional, state-of-the-art and future thermal building insulation materials and solutions - properties, requirements and possibilities', *Energy and Buildings*, **43**(10), 2549 – 2563, (2011).

[15] U. Junker, *Configuration.*, Chapter 24 of Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA, 2006.

[16] K.J. Lee, H.W. Kim, J.K. Lee, and T.H. Kim, 'Case-and constraint-based project planning for apartment construction.', *AI Magazine*, **19**(1), pp. 13–24, (1998).

[17] B. Medjdoub and B. Yannou, 'Separating topology and geometry in space planning', *Computer-Aided Design*, **32**(1), 39 – 61, (2000).

[18] U. Montanari, 'Networks of constraints: Fundamental properties and applications to picture processing', *Information Sciences*, **7**(0), 95 – 132, (1974).

[19] L. Pérez-Lombard, J. Ortiz, and C. Pout, 'A review on buildings energy consumption information', *Energy and Buildings*, **40**(3), 394 – 398, (2008).

[20] C. Prud'homme and JG. Fages, 'An introduction to choco 3.0 an open source java constraint programming library', in *CP Solvers: Modeling, Applications, Integration, and Standardization. International workshop.*, Uppsala Sweden, (2013).

[21] Christian Schulte, *Programming Constraint Services*, volume 2302 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2002.

[22] Christian Schulte and Mats Carlsson, 'Finite domain constraint programming systems', *Handbook of constraint programming*, 495–526, (2006).

[23] Christian Schulte and Gert Smolka. Finite domain constraint programming in oz. a tutorial., 2000.

[24] Christian Schulte, Guido Tack, and Mikael Z. Lagerkvist. Modeling and programming with gecode, 2010.

[25] S Shikder, A Price, and M Mourshed, 'Interactive constraint-based space layout planning', *W070-Special Track 18th CIB World Building Congress May 2010 Salford, United Kingdom*, 112, (2010).

[26] Timo Soininen, Juha Tiihonen, Tomi Männistö, and Reijo Sulonen, 'Towards a general ontology of configuration', *Artif. Intell. Eng. Des. Anal. Manuf.*, **12**(4), 357–372, (September 1998).

[27] Élise Vareilles. Paul Gaborit. Michel Aldanondo. Sabinne Carbonnel. Laurent Steffan, 'Cofiade constraints filtering for aiding design', in *Actes des neuviemes Journées Francophones de Programmation par Contraintes*, Toulouse France, (2012).

[28] E. Vareilles, A. F. Barco, M. Falcon, M. Aldanondo, and P. Gaborit, 'Configuration of high performance apartment buildings renovation: a constraint based approach', in *Conference of Industrial Engineering and Engineering Management (IEEM)*. IEEE., (2013).

[29] Dong Yang, Ming Dong, and Rui Miao, 'Development of a product configuration system with an ontology-based approach', *Computer-Aided Design*, **40**(8), 863 – 878, (2008).

[30] M. Zawidzki, K. Tateyama, and I. Nishikawa, 'The constraints satisfaction problem approach in the design of an architectural functional layout', *Engineering Optimization*, **43**(9), pp. 943–966, (2011).